REMARKS

Claims 1, 3-22, 24-43 and 45-53 remain pending in the application. Reconsideration is respectfully requested in light of the following remarks.

Section 103(a) Rejection:

The Examiner rejected claims 1, 3-22, 24-43 and 45-53 under 35 U.S.C. § 103(a) as being unpatentable over Erickson et al. (U.S. Patent 6,851,089) (hereinafter "Erickson") in view of Baentsch et al. (U.S. Patent 6,772,171) (hereinafter "Baentsch"). Applicants respectfully traverse this rejection for at least the following reasons.

Regarding claim 1, contrary to the Examiner's assertion, Erickson in view of Baentsch fails to teach or suggest the client device receiving a message in the data representation language from a service device in the distributed computing environment prior to generating a computer programming language object, wherein the message includes the data representation language representation of the object. The Examiner cites column 26, lines 15-20 of Erickson. However, the Erickson's wrapper builder application, described at the Examiner's cited passage, does not teach or suggest, even if combined with Baentsch's smartcard-based persistent object generation system, a *message* including a data representation language representation of a computer programming language object, let alone a client device *receiving* such a message *from a service device in a distributed computing environment*. Erickson teaches that his wrapper builder and wrapper execution applications load wrappers using serialization *locally from disk*. The Examiner also relies on Baentsch, citing column 7, lines 46-50 and column 8, lines 26-36. However, the cited passages of Baentsch do not have anything to with receiving a data representation language message from a service device in a distributed computing environment, contrary to the Examiner's contention. At the cited passage, Baentsch describes a sequence of instruction code executed in order to create a new object within the non-persistent memory of a smartcard device. Baentsch has nothing to do with clients and services exchanging data representation language messages including

data representation language representations of objects within a distributed computing environment. Instead, Baentsch is concerned with "creation of persistent and temporary objects and the transfer of temporary objects between multiple invocations of an application" (Baentsch, column 3, lines 65-67).

The Examiner's combination of Erickson and Baentsch would not result in a system including receiving a message in the data representation language from a service device in a distributed computing environment, where the message includes the data representation language representation of a computer programming language object. First of all, as shown above, the Examiner's reliance on Baentsch regarding receiving a data representation language message from a service device is clearly misplaced. Baentsch's system involves object generation and persistence within a single smartcard environment. Even a cursory reading of Baentsch indicates that Baentsch does not mention anything, nor does his system involve, a client receiving a data representation language message from a service device in a distributed computing environment.

Since, as shown above and admitted by the Examiner, Erickson fails to teach or suggest receiving a data representation language message from a service device, and since Baentsch also fails to teach or suggest receiving a data representation language message from a service device, any combination of Erickson and Baentsch clearly fails to teach or suggest receiving a data representation language message from a service device.

Furthermore, Erickson in view of Baentsch fails to teach or suggest deleting, in response to a user terminating accessing the client device, the computer programming language object that was generated from the data representation language representation of the object included in the message received from the service device in the distributed computing environment. The Examiner relies on Baentsch, citing column 8, lines 63-66 where Baentsch describes automatically deleting transient or temporary objects when "the applet is deselected or when the power is shut off." The Examiner's reasoning appears to be that this teaching in Baentsch would suggest deleting upon termination the wrapper object in Erickson that the Examiner

corresponds to the computer programming language object of Applicants' claim 1. **However, such a modification would be directly counter to the explicit teachings of Erickson.** Erickson teaches that his wrapper should persist for subsequent use (see, e.g., Abstract, last sentence). Erickson's entire system is directed toward the design and creation of wrapper files for *subsequent* use when accessing web sites (Erickson, FIG. 15; column 2, lines 58-65; and column 27, lines 9-59). Erickson specifically teaches that his wrapper object, which the Examiner equates to the computer programming language object of Applicants' claim, should persist for subsequent use (e.g., persist until a future accessing – See, Abstract, last sentence). Moreover, the Examiner is overlooking that Baentsch also teaches that "persistent objects are objects which keep their state between different hardware activations." Thus, the passage of Baentsch relied on by the Examiner is clearly referring to deleting temporary or transient objects, but the wrapper objects in Erickson are not temporary or transient objects. Thus, this teaching in Baentsch would not apply to the wrapper objects in Erickson. In fact, Baentsch's teaching that persistent objects should **not** be deleted actually reinforces Applicants' argument that both references actually teach away from deleting, in response to a user terminating accessing the client device, the computer programming language object that was generated from the data representation language representation of the object included in the message received from the service device in the distributed computing environment.

If one were to modify Erickson according to the teaching of Baentsch, the combination would not result in a system that involved deleting Erickson's wrapper objects. Instead, the combination would still persist Erickson's wrappers as taught by both Erickson and Baentsch but would delete other, temporary or transient objects (which would not meet the other limitations of Applicants' claim), as taught by Baentsch (and relied on by the Examiner).

Thus, the Examiner's combination of cited art clearly fails to teach or suggest all the limitations of Applicants' claims.

Moreover, the Examiner has failed to provide a proper motivation to combine Erickson and Baentsch. The Examiner gives several motivations for modifying Erickson as taught by Baentsch, including: "allows cleanup operations to be performed and afterwards selects the new application", "supports the maintenance of temporary objects between multiple invocations of an application during one session", "supports the maintenance of temporary object between multiple invocations of an application during one session without using persistent memory" and "enables an application to ensure its integrity in case of an unexpected power down." However, none of the Examiner's stated motivations would motivate one to modify Erickson's wrapper builder application to include the smartcard based temporary object deletion method taught by Baentsch.

Firstly, Baentsch's system works on, and solves problems inherent with, specific smartcard-style environments while Erickson's system works on "network-connected PC or workstation supporting Java applications and the standard protocols and conventions of the World Wide Web" (Erickson, column 2, lines 58-65). The problems for which Baentsch's system is designed to solve simply do not exist in the computing environment required for Erickson's system. For instance, Erickson states that in the preferred embodiment, "the application provides a graphic design environment in which a wrapper can be created from operations and links" and that "the application also provides a visual run and debug environment that operates in conjunction with the graphical design environment." Erickson's is intended to provide "a software layer that functions to provide automated access to semistructured information", such as to "enable information stored on a Web page to be accessed through an SQL query" (Erickson, column 2, lines 26-38). In contrast, Baentsch is concerned with "[o]bject-based applications running inside extremely resource-constrained execution environments, such as smartcards" (Baentsch, column 2, lines 14-20 and lines 45-47). Thus, Baentsch system works in "extremely resource-constrained execution environments" while Erickson's works "network connected PC[s] or workstation[s]" and that provide "a graphical design environment", "a visual run and debug environment" and "a Web viewer with which the user can browse Web sites for information of interest while creating a wrapper"

(Erickson, column 2, line 66 – column 3, line 2; column 3, lines 14-19, lines 36-41). **Thus, Erickson actually teaches away from the Examiner's proposed combination.**

Additionally, both **Erickson and Baentsch teach away** from the Examiner's proposed combination regarding modifying Erickson to include deleting the computer programming language object in response to a user terminating accessing the client device. As noted above, Erickson's entire system is directed toward the design and creation of wrapper files that should persist for *subsequent* use when accessing web sites (Erickson, FIG. 15; Abstract; column 2, lines 58-65; and column 27, lines 9-59). Moreover, the Examiner is relying on Baentsch's teaching the automatic deletion of temporary objects. However, Baentsch specifically teaches that persistent objects, such as the wrappers in Erickson, should **not** be deleted. Thus, both **Erickson and Baentsch teach away** from modifying Erickson to delete wrapper files and thus from their combination. As the Examiner is aware, where the individual reference teach away from their combination, there is no motivation or suggestion to combine.

In addition, as shown above, modifying Erickson to delete the wrapper files in response to a user terminating access, as proposed by the Examiner, would clearly render Erickson's system unsatisfactory for its intended purpose (e.g., design and creation of wrapper files for *subsequent* use) and change the principle of operation. As stated in M.P.E.P. § 2143.02, if the "proposed modification would render the prior art invention being modified unsatisfactory for its intended purpose" or if the modification "would change the principle of operation" then there is no suggestion or motivation to make the proposed modification.

Furthermore, even if the combination was properly motivated (which it is not), the Examiner's combination of Erickson and Baentsch fails to teach all the limitations of Applicants' claim. As illustrated above, Erickson in view of Baentsch does not teach or suggest the client device receiving a message in the data representation language from a service device in the distributed computing environment prior to generating a computer

programming language object, wherein the message includes the data representation language representation of the object.

The Examiner has clearly failed to provide a *prima facie* rejection. The Examiner's reliance on Baentsch is misplaced, the combination of Erickson and Baentsch does not teach or suggest all the limitations of Applicants' claim and there is no motivation to combine Erickson and Baentsch. For at least the reasons above, the rejection of claim 1 is not supported by the cited art and removal thereof is respectfully requested. Similar remarks apply to claims 22 and 43.

Regarding claim 10, Erickson in view of Baentsch fails to teach or suggest **a client device receiving a message in a data representation language from a service device in the distributed computing environment, wherein the message includes a data representation language representation of a computer programming language object**. Please refer to Applicants' remarks above regarding the rejection of claim 1, as they apply to the rejection of claim 10 as well.

Additionally, Erickson in view of Baentsch does not teach or suggest **generating a computer programming language object from a data representation language representation of the object if it is determined that the user has access rights to the computer programming language object**. The Examiner cites Baentsch at column 6, lines 56-67 and column 7, line 65 – column 8, line 12. However, the cited passages do not mention, no have anything do with, generation a computer programming language object from a data representation language representation of the object if it is determined that the user has access rights to the computer programming language object. Instead, the cited passages describe how Baentsch's system begins execution when a smartcard is inserted into a card reader (Baentsch, column 7, lines 57-67). The Examiner is apparently misinterpreting Baentsch teaching regarding allowing access, within a smartcard environment, of stored objects without activating the virtual machine. Baentsch is describing allowing access to the memory unit of a smartcard without actually activating the virtual machine (VM). Thus, the cited portion of Baentsch, even if

combined with Erickson, clearly does not describe the generation of any computer programming language object because Baentsch specifically states that the virtual machine (required for any Java code to execute) is not activated.

Moreover, Erickson in view of Baentsch does not teach or suggest **determining if the user has access rights to the computer programming language object**. The Examiner does not actually cite any portion of cited art regarding this limitation of claim 10. Presumably, the Examiner is relying on Baentsch at column 6, lines 56-67 and column 7, line 65 – column 8, line 12. However, the Examiner's reliance on Baentsch is misplaced. The cited passages of Baentsch describe accessing the memory unit of a smartcard environment without activating the virtual machine. Nowhere does Baentsch mention anything about determining whether a user has access rights to a computer programming language object. Baentsch, even if combined with Erickson, is not concerned with determining whether a user has access rights. Baentsch is concerned with a new method of co-existing temporary and persistent objects in "extremely resource-constrained execution environments."

Furthermore as noted above regarding the rejection of claim 1, the Examiner has also failed to provide a proper motivation to combine Erickson and Baentsch. In fact, the Examiner provided no discussion whatsoever of a motivation to combine in regard to the specific limitations of claim 10.

Thus, for at least the reasons above, the rejection of claim 10 is not supported by the cited art and removal thereof is respectfully requested. Similar remarks apply to claims 32 and 47.

Regarding both § 103 rejections, Applicants also assert that numerous ones of the dependent claims recite further distinctions over the cited art. However, since the rejection has been shown to be unsupported for the independent claims, a further discussion of the dependent claims is not necessary at this time.

CONCLUSION

Applicants submit the application is in condition for allowance, and prompt notice to that effect is respectfully requested.

If any fees are due, the Commissioner is authorized to charge said fees to Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C. Deposit Account No. 501505/5181-47300/RCK.

Respectfully submitted,

/Robert C. Kowert/
Robert C. Kowert, Reg. #39,255
Attorney for Applicants

Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C.
P.O. Box 398
Austin, TX 78767-0398
Phone: (512) 853-8850

Date: April 26, 2007